Services

In angular services are reusable classes which can be injected in components when it's needed.

Using a separate service keeps components lean and focused on supporting the view, and makes it easy to unit-test components with a mock service.

Employee.ts

```
export class Employee
{
   id: number;
   name: string;
   salary: number;
   dateOfJoin: Date;
   constructor(id: number, name: string, sal: number, doj: Date)
   {
      this.id = id;
      this.name = name;
      this.salary = sal;
      this.dateOfJoin = doj;
   }
}
```

Step2: Creating Injectable service: We are creating an Injectable service using angular @Injectable decorator. When using @Injectable should use **parenthesis ( ).** The **@Injectable()** decorator tells TypeScript to emit metadata about the service. The metadata specifies that Angular may need to inject other dependencies into this service.

Employee.service.ts

```
import { Injectable } from '@angular/core';
import { Employee } from "./employee"

@Injectable()
export class EmployeeService {
   getEmployees(): Employee [] {
      let employees: Employee[] = [
         new Employee(1, "Raj Kumar", 11000.12345, new Date(2016, 0, 20)),
         new Employee(2, "Sudheer Reddy", 12000, new Date(2016, 0, 22)),
```

```
            new Employee(3, "Kiran Rao", 13000, new Date(1983, 12, 32)),
            new Employee(4, "Kishor Kumar", 14000, new Date(1983, 5, 3)),
            new Employee(5, "Usha Rani", 12000, new Date(1985, 5, 31)),
            new Employee(6, "Sai Kiran", 12300, new Date(1986, 6, 36)),
            new Employee(7, "Jacob John", 11000, new Date(1987, 7, 33)),
        ]
        return employees;
    }
}
```

Step3: Injecting Service:

Instead of using the **new**, you'll add two lines.

- Add to the component's **providers** metadata.
- Add a constructor that also defines a **private property**.

Note: For **Async services** we can use **promises** which is an asynchronous technique. Instead of returning the data from the service we will return the data as a promise.

employees.component.ts

```
import { Component } from '@angular/core';

import { Employee } from './employee';
import { EmployeeService } from "./employee.service"

@Component({
    moduleId: module.id,
    selector: 'employees',
    providers: [EmployeeService],
    template: `
        <style>
        table, th, td {
            border: 1px solid black;
        }
        </style>
        <table>
            <tr *ngFor="let emp of employees">
                <td>{{emp.id}}</td>
```

```
            <td>{{emp.name | uppercase}}</td>
            <td>{{emp.salary | currency:'USD':true:'4.2-3'}}</td>
            <td>{{emp.dateOfJoin | date:'yMMMdjms'}}</td>
        <tr>
    </table>
`
})
export class EmployeesComponent {
  employees: Employee[]
  constructor(private empService: EmployeeService) //EmployeeService is Injected here…
  {
    this.employees = empService.getEmployees();
  }
}
```

Note: We can either register a provider within an NgModule or in application components:

Service Using a Service

Step 1: Write a New Service.

Logger.service.cs

```
import { Injectable } from '@angular/core';


@Injectable()
export class Logger {
  logs: string[] = []; // capture logs for testing
  log(message: string) {
    this.logs.push(message);
    console.log(message);
  }
}
```

Step 2: Using Service within a Service.

Employee.server.ts

```
import { Injectable } from '@angular/core';
import { Employee } from "./employee"
import { Logger } from "./logger.service"
```

```
@Injectable()
export class EmployeeService {
   constructor(private logger: Logger) { this.logger = logger; }
   getEmployees(): Employee[] {
      this.logger.log('Fetching employees...');
      let employees: Employee[] = [
         new Employee(1, "Raj Kumar", 11000.12345, new Date(2016, 0, 20)),
         new Employee(2, "Sudheer Reddy", 12000, new Date(2016, 0, 22)),
         new Employee(3, "Kiran Rao", 13000, new Date(1983, 12, 32)),
      ]
      return employees;
   }
}
```

Step 3: To Register a Logger Service as provider in an NgModule. Once registered here it can be used in any component.

App.module.ts:

```
import { Logger } from './logger.service'

@NgModule({
 imports:      [ BrowserModule, FormsModule ],
 declarations: [AppComponent, EmployeesComponent, AgePipe, SalaryPipe ],
 bootstrap: [AppComponent],
 providers: [Logger]
})
```

$http Services

As the standards are going into next levels these days we are making use of HTTP calls to external API's. To do this angular has built-in HTTP library.

There are two ways for processing this asynchronous operations

- **Using Promises**
- **Using Observables**

It's recommended to use **$http** calls within wrapped services instead of components which gives us more flexibility to our application. To use this service we need to just import **(Http)** it and use it.

Http (Hyper Text Transfer Protocol) is used to communicate with the server, where most of the browsers supports two Http based requests (**XHR and JSONP**), and few browsers also supports **Fetch**

- **XML Http Request (XHR):** This request fetches data from server without refreshing page again.
- **JSONP:** This request is mainly used for cross domain requests i.e., calling a service of one domain from another domain.
- **Fetch:** This request fetches with promises.

Step1: Configure the application

File: **app.module.ts**

Import Http Service form angular library, **HttpModule** is necessary for making http calls where as **JSONP** is not imported here because it's not important for **plain Http**

```
import { HttpModule } from '@angular/http';
```

Now register the providers to **root NgModule**

```
import { HttpModule } from '@angular/http';


@NgModule({
   imports: [BrowserModule, HttpModule],
   declarations: [AppComponent],
   bootstrap: [AppComponent]
})
export class AppModule { }
```

Step2: Creating model for Employee

File: **app.models.ts**

```
export class Employee {
   EmpId: number;
   EmpName: string;
   EmpSalary: number;
   constructor() {
   }
}
```

Step3: Create a service

File: **app.EmpService.ts**

```
import {   } from '@angular/http';
import { Injectable } from '@angular/core';
import 'rxjs/add/operator/map';
import { Observable } from 'rxjs/observable';
import { Employee } from './app.models';

@Injectable()
export class EmpService {
   employees: Employee[];
   constructor(private http: Http) { }
   GetEmployees(): Observable<Employee[]> {
      return this.http.get("/api/EmployeeAPI").map((res: Response) => res.json());
   }
}
```

Here we have imported

- **Injectable:** which is used to create an injectable service, so that it can be injected anywhere.
- **Observable:** Used for processing Async operations
- **rxjs/add/operator/map:** This map operator applies a function to the items emitted by observable.
- **Http, Response,RequestOptions,Headers:** These are imported from angular Http libray
    1. **Http:** for making http request
    2. **Response:** The response object which is returned from the request.
    3. **Headers:** Headers object can be configured.
    4. **RequestOptions:** This holds all the configurations and send it along with the request as an arugument.

Get Request

Step4: Creating Component that calls **EmpService** methods

File: **app.component.ts**

```
import { Component, OnInit} from '@angular/core';
import { EmpService } from './app.EmpService';
import { Employee, Department } from './app.models';
@Component({
```

```
  templateUrl: './EmpTemplate.html',
  providers: [EmpService]
})
export class AppComponent implements OnInit {
  employees: Employee[];
  constructor(private empService: EmpService) { }
  ngOnInit() {
    this.GetEmployees();
  }
  GetEmployees() {
    this.empService.GetEmployees().subscribe(emps => this.employees = emps);
  }
}
```

File: **EmpTemplate.html**

```
<table class="table table-striped table-bordered" style="width:50%">
    <thead>
      <tr>
        <th>Employee name</th>
        <th>Employee Salary</th>
      </tr>
    </thead>
    <tbody>
      <tr *ngFor="let emp of employees">
        <td>{{emp.EmpName}}</td>
        <td>{{emp.EmpSalary}}</td>
      </tr>
    </tbody>
  </table>
```

File: **main.ts**

```
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { AppModule } from './app.module';
platformBrowserDynamic().bootstrapModule(AppModule);
```

Employee name	Employee Salary
Suresh	12000
Muni	12500
Phani	17500
kavitha	15001
Yeswanth	12000

Get By Param: We can pass params with a different syntax where the parameters will be wrapped in between flower braces ${ } and the const URL must be framed with in ` `

Ex: const url = `${'Url'} / ${param}`

Add this code in **app.EmpService.ts**

```
GetEmployee(empId: any): Observable<Employee> {
    const url = `${'/api/ManageEmployee'}/${empId}`;
    return this.http.get(url).map((res: Response) => res.json());
}
```